2011年7月14日　統計数理研究所　オープンハウス

GridRPCを用いたRによる並列計算環境

中野　純司　データ科学研究系、データ同化開発研究センター　教授

【はじめに】

Rは統計解析とグラフィックスのためのフリーのソフトウェア環境である。現在の計算機環境においては、ネットワークで結ばれたローカルあるいはリモートにある複数の計算機（スーパーコンピュータも含む）上で、Rを同時に簡単にかつ効率的に利用できることが望ましい。この目的を実現するために、われわれはRでNinf-GのGridRPCミドルウェアを利用できるようにした。

【GridRPC】

GridRPCは、主としてグリッドのための遠隔手続き呼び出し（Remote Procedure Call, RPC）によるプログラムモデルである。Ninf-GとNetsolveがGridRPCミドルウェアとしては有名である。他のGridRPCミドルウェアとしては、GridSolve、DIET、OmniRPCなどがある。われわれは、RでGridRPC機能を実現するためにNinf-Gを利用する。Ninf-Gは、Globus Toolkitを利用するGridRPCの実現であるが、Gridミドルウェアがない場合、セキュアシェル (SSH) で接続することもできる。

【RGridRPCの概略】

RGridRPCは Embedded R を利用して実現されたRのパッケージである。Rの1つのプロセスは、ハンドルの生成で始まり、その終了で終わる。

【RGridRPCの基本的関数】

・クライアント初期化および終了化関数

```
.grpc_initialize(config_file), .grpc_finalize()
```

・ハンドル関数

```
.grpc_function_handle_init(hostname)
.grpc_function_handle_default()
.grpc_function_handle_destruct(handle)
```

・同期関数

```
.grpc_call(handle,fun,...)
```

・非同期関数

```
.grpc_call_async(handle,fun,...)
.grpc_probe(session), .grpc_wait(session)
```

【RGridRPCの“snow”に似た関数】

“snow” はRにおける並列計算のための標準的なパッケージである。

・クライアント初期化および終了化関数

```
GRPCmake(hostname), GRPCstop(handle)
```

・同期関数

```
GRPCclusterApply(handle,x,fun,...),GRPCevalq(handle,expr)
GRPCexport(handle,names), GRPCcall(handle,fun,...)
```

・非同期関数

```
GRPCcallAsync(handle,fun,...), GRPCprobe(section)
GRPCwait(section)
```

【インストール手順】

```
# download
$ wget \
   http://prs.ism.ac.jp/RGridRPC/RGridRPC_0.10-302.tar.gz
# create R_LIBS_USER directory
$ R -q -e 'dir.create(Sys.getenv("R_LIBS_USER"),rec=T)'
# install
$ R CMD INSTALL RGridRPC_0.10-302.tar.gz
```

Gridミドルウェアを使用するときは、それぞれの計算ノードにNinf-Gをインストールし、NG_DIR環境変数を適切なものにする。それからRGridRPCをインストールする。

```
# Using Grid
$ NG_DIR=/opt/ng R CMD INSTALL RGridRPC_0.10-302.tar.gz
```

【RGridRPCセットアップ】

RGridRPCは、コンフィギュレーションファイルとしてカレントディレクトリの`client.conf`を読みこむ。RGridRPCでは双方向通信が必要である。したがって、クライアントは`client.conf`によってそのホスト名をサーバーに知らせなければならない。RGridRPCは情報記述のためにNRF (Ninf-G Remote Information File) を利用する。

【実行例】

パッケージpvclustの単独実行例

```
$ R CMD BATCH pvclust1.R
> library(pvclust)
> library(MASS)
> data(Boston)
> nboot<-6000
> set.seed(1)
> system.time(boston <- pvclust(Boston, nboot=nboot))
Bootstrap (r = 0.5)... Done.
Bootstrap (r = 0.6)... Done.
Bootstrap (r = 0.7)... Done.
Bootstrap (r = 0.8)... Done.
Bootstrap (r = 0.9)... Done.
Bootstrap (r = 1.0)... Done.
Bootstrap (r = 1.1)... Done.
Bootstrap (r = 1.2)... Done.
Bootstrap (r = 1.3)... Done.
Bootstrap (r = 1.4)... Done.
user system elapsed
497.263 0.200 497.480
```

パッケージpvclustの並列実行例

```
$ eval `ssh-agent`
Agent pid
$ ssh-add .ssh/id_rsa
Enter passphrase for .ssh/id_rsa:
Identity added: .ssh/id_rsa (.ssh/id_rsa)
$ R CMD BATCH pvclust2.R
> library(pvclust)
> library(MASS)
> data(Boston)
> library(RGridRPC)
> library(snow)
> # Replace a snow function by an RGridRPC function
> parLapply<-function (cl, x, fun, ...){ docall(c,
+ GRPCclusterApply(cl, splitList(x, length(cl)), lapply,
+ fun, ...)) }
> # Make 30 workers on 3 systems
> cl<-GRPCmake(c(rep("prs.ism.ac.jp",2),
+ rep("prcs.ism.ac.jp",12),
+ rep("prds.ism.ac.jp",16)))
> nboot <- 6000
> # Set seeds
> dummy<-GRPCclusterApply(cl,1:length(cl),set.seed)
> system.time(boston.pv <- parPvclust(cl, Boston,
+ nboot=nboot))
Multiscale bootstrap... Done.
user system elapsed
0.172 0.036 21.432
> GRPCstop(cl)
```